DOCUMENT RESUME

ED 151 018 IR 005 742

AUTHOR Francis, Larry; And Others
TITLE Selected Characteristics of TUTOR Programs Produced in ARPA-Sponsored PLATO Projects.
INSTITUTION Illinois Univ., Urbana. Computer-Based Education Lab.
SPONS AGENCY Advanced Research Projects Agency (DOD), Washington, D.C.; National Science Foundation, Washington, D.C.
PUB DATE 78
CONTRACT DAHC-15-73-C-0077; USNSF-C-723
NOTE 37p.

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Oriented Programs; Computer Programs; Instructional Improvement; Military Training; Program Development; *Programing; Program Planning

ABSTRACT

This document reports on a study of several characteristics of the Computer Based Education (CBE) lessons written at military training sites whose CBE projects were sponsored by the Defense Advanced Research Projects Agency (ARPA). The purposes of the study were (1) to demonstrate the potential of computer-based scanning of CBE programs as a means for extracting information about the programming techniques used; (2) to suggest the use of computer-scanning techniques to aid the management of CBE development efforts; and (3) to provide additional information about the products of CBE lesson development projects conducted under ARPA sponsorship. The lesson characteristics selected for examination were chosen on the basis of: (1) the ease and reliability of their measurement and (2) their importance in CBE. Most data are reported on a site-by-site basis. (Author/DAG)

SELECTED CHARACTERISTICS OF TUTOR PROGRAMS PRODUCED IN ARPA-SPONSORED PLATO® PROJECTS

Larry Francis
Joseph A. Klecka
A. Lynn Misselt

COMPUTER-BASED EDUCATION RESEARCH LABORATORY
UNIVERSITY OF ILLINOIS, Urbana-Champaign

This research was supported in part by Advanced Research Projects Agency of the Department of Defense under U.S. Army Contract DAHC-15-73-C-0077 and indirectly by the National Science Foundation (USNSF C-723).

ACKNOWLEDGMENTS

The authors wish to thank Kathy Geissler and Julie Garrard for their patience, carefulness, and thoroughness in data collection activities. We appreciate the editorial assistance given by Elaine Avner, and we thank R. A. Avner for the many constructive suggestions he gave and for the encouragement he offered throughout the study.

TABLE OF CONTENTS

INTRODUCTION

Background

This document reports on a study of several characteristics of the Computer Based Education (CBE) lessons written at military training sites whose CBE projects were sponsored by the Defense Advanced Research Projects Agency (ARPA). The purposes of the study were:

1. to demonstrate the potential of computer-based scanning of CBE programs as a means for extracting information about the programming techniques used,

2. to suggest the use of computer-scanning techniques to aid management of CBE development efforts, and

3. to provide additional information about the products of CBE lesson development projects conducted under ARPA sponsorship.

The scope of the study was limited to include only a few of the more interesting and potentially useful parameters that can be easily measured by computer-scanning techniques. The lesson characteristics selected for examination were chosen on the basis of (a) the ease and reliability of their measurement and (b) their importance in CBE. Many other characteristics could have been included, but the number studied is sufficient to illustrate the potential of the technique.

Most data are reported on a site-by-site basis. The names of the various sites are presented, not for the purpose of pointing out strengths and deficiencies, but in order to allow readers who are familiar with the materials produced at a given site to relate their intuitive perceptions of lesson characteristics to the actual quantities we have measured.

In general, the aim of the report is to illustrate the techniques of computer-scanning and to identify important relationships among categories of lesson characteristics. It is not our intent to critique the efforts of individual sites.

The lessons examined here were also studied in investigations of author language proficiency (Francis, 1976a) and the use of peripheral devices connected to the terminal (Francis, 1976b). The general technique of using the computer to scan or search through the source code for the presence of certain target "character strings" was applied in both the study of language proficiency and the investigation of terminal peripheral device usage.

Although the quantities examined in this study and the data collection routines used to measure them are those which are provided by the PLATO system, the techniques used here can be applied to other CBE systems which support similar features. Nevertheless, we assume that most readers will have some familiarity with the PLATO-oriented terminology for various features and measured quantities. Because documentation of features of the PLATO system and its authoring language is readily available (cf. Wood, 1975; Lyman, 1977; Sherwood, 1977), details of these features are not presented here.

Data Collection Procedures

The first step in the data collection process was to determine the nature and number of the lessons produced at each of the ARPA-sponsored sites. Sites which developed only a small number of lessons (five or fewer) or which produced only lesson fragments were excluded. Next, the usage data for each of the lessons was examined and any lesson having five or fewer hours of usage during the previous year was discarded. The latter technique insured that data storage files, note files, incomplete lessons, and many other "non-

lessons" would be eliminated from the target group. Finally, the lists of site-developed programs were corrected "by hand" so that some genuine but seldom-used lessons could be included.

The data were collected by means of the PLATO system's "X-search" option. This feature allows the investigator to specify a word, phrase, or set of letters (i.e., a "character string") and let the computer locate all occurrences of that string. For example, if the investigator wishes to find how many comments (i.e., statements describing the nature and purpose of a piece of TUTOR coding) the author has placed in a program, he enters the X-search option and specifies all the possible indicators of a comment. In TUTOR, the PLATO system's authoring language, comments can be indicated by two consecutive dollar signs in the "tag field" of a TUTOR statement, a "c" at the beginning of a line followed by a space (i.e., "c "), or by an asterisk at the beginning of a line. Upon locating one or more of these character strings in a line of the program, the computer displays that line on the terminal's screen. The investigator can then determine by looking at the entire line whether the occurence of "c ", say, is at the beginning (and hence actually indicates a comment) or is only a part of some other coding or text. Decisions of this kind can be made rather easily by clerical staff having minimal training.

Sites

The ARPA authors had been taught or given suggestions about various lesson characteristics during their TUTOR training at the Computer-based Education Research Laboratory (CERL) and through consultation with the CERL staff. In most cases only general guidelines were given to the ARPA authors: for example, they were told to document any programming which was unusual, complex, or which interacted with coding in other

files. The meaning of "unusual", "complex", or "interacted with" was illustrated by examples, but was not further delineated. In general, the local environment of the author was more a determinant of his behavior than was the training he received at CERL. Those environments did, of course, vary from site to site, and it is for this reason that a variety of sites were included in the study (see Appendix I for descriptions of the sites). Furthermore, reporting the data on a site-by-site basis serves to illustrate the variation in sites' goals and standards.

Unfortunately, it was not possible to apply exactly the same criteria in selecting the programs/lessons to be examined from each site, because of variations in site goals, methods, and types of materials produced. Furthermore, the computer programs selected for examination from some sites included non-instructional routines (e.g., routers and drivers) and/or programs written solely for research purposes as well as those written for instruction.

Table 1 summarizes the major characteristics of the programs examined at each of the sites. Readers should refer to Table 1 to facilitate interpretation of results presented later in the report.

Table 1

Characteristics of TUTOR Programs Produced at ARPA Sites

Site[a]	Student Tested	Tutorial Lessons	Some Simulations	Incomplete Lessons	% Lessons[b] for Course	Res/Op[c] Orientation	Lessons[d] Only
CHA	X	X		No	100%	Op	No
ABE	X	X		No	100%	Op	No
SHP	X	X	X	10%	90%	Op	No
NTC	X	X		No	20%	Res	Yes
N.I.	X		X	No	100%	Op	Yes
ORL	X	X		No	0%	Res	Yes
MAX		X		10%	0%	Res	Yes
ARI	X	X		20%	0%	Res	No

[a] For convenience, the site names are abbreviated throughout the report. See Appendix I for a more complete identification of the sites.

[b] Percentage of selected programs designed to meet an existing training need at the site (i.e., percentage of programs which are "non-research")

[c] Objectives of sites are either research (Res) or operational (Op) in nature

[d] Lessons only - tallied as "Yes" if the programs examined were all intended as teaching materials (i.e., lessons); tallied as "No" if programs also included drivers, routers, and data management routines.

Note. All data cited in this report were taken from the lessons as they appeared in June 1976. At that point nearly all sites had been in operation for two to four years and had established patterns of behavior.

SPECIFIC CHARACTERISTICS

Documentation

We use the term "documentation" broadly to indicate both the naming of variables and the commenting of lines or segments of programming. On the PLATO system, documentation can be implemented via notes off-line (on printed copies of the lesson or in notebooks) or via on-line comments. Because a poll of ARPA authors indicated little if any off-line documentation, the study of on-line documentation herein can be viewed as being essentially comprehensive.

The need for documentation is undisputed. Its quantity should be proportional to the complexity of the coding used: the more complex the program, the greater the need for documentation. Because many of the lessons written by ARPA authors do not use complex programming, it might be expected that these lessons would not require extensive documentation. Nevertheless, good practice dictates that some documentation be provided even for relatively straightforward programming.

The level of documentation was measured by examining: (a) the number of on-line comments of all kinds, and (b) the use of defined variables.

On-line comments. Although comments can be placed in PLATO lessons in three ways (see Appendix II), we did not distinguish among the three comment "formats" in making our counts. As lines of the TUTOR code potentially containing comments were found and displayed by the computer, they were tallied as comments if they contained any sort of information that would be useful to the author or a later "caretaker" of the programs.

Table 2 summarizes the rates of use of comments in ARPA-sponsored TUTOR programs.

Table 2

Use of Comments in Programs Developed at ARPA Sites

Site	N[a]	% Lessons with Comments	Total # Comments	Average # Comments/ Lesson
CHA	37	24	31	0.8
ARI	20	90	313	15.6
MAX	7	0	0	0.0
ORL	17	31	57	3.4
ABE	22	73	192	8.7
SHP	66	50	170	2.6
NTC	15	13	12	0.8
N.I.	12	100	172	14.3

[a] N = total number of lessons examined at each site

It is difficult to gauge the adequacy of the on-line comments used by authors at the various sites. For example, it is possible that the Chanute programming is so straightforward that persons responsible for making revisions can do so readily even with only an average of .8 commented lines per lesson. On the other hand, the nature of the programming in lessons produced at the Naval Personnel Research and Development Center (NPRDC) North Island Site (N.I.) might be so complex that even an average of 14.3 comments per lesson is insufficient from the viewpoint of the person charged with making revisions. Not having attempted to unravel the intricacies of the coding used in each program, we are hesitant to pass judgment on the adequacy of the comments provided by the original authors. Nevertheless, on the basis of our experience with inadequate documentation, we are sorry to find so many lessons with no comments at all (see the

second column of data in Table 2). Perhaps the best use of the results in Table 2 is as a benchmark against which managers of courseware development projects can gauge lessons produced under their control. These data will be most useful for those managers who are familiar with the lessons written at ARPA sites and hence are able to judge the relevancy of these results for their own situation.

Named variables. The computer storage locations or variables available for use in programs in the TUTOR language can be accessed in their "primitive" form (e.g., n1,n2,...,n150: v1,v2,...,v150) or they can be given meaningful names of up to seven characters in length (e.g., n1=index, n2=count, v3=root, etc.). Because unique names can be associated with the primitive variables by use of a -define- command, such variables are said to have been "defined". There are several advantages to using defined variables, but the one of greatest current interest is that complex uses of meaningfully-named variables are much easier to decipher than those using variables in their primitive, un-named form. The use of defined variables, therefore, is an important method of documentation.

All programs selected from each site were searched for the presence of -define- commands. Unlike "comments" which can and should be distributed throughout a program, -define- commands are typically used only once per lesson. A single -define- can be used to associate names with practically any number of variables. Hence, the absence of a -define- command is quite telling because it indicates that no named variables were used. On the other hand, the mere presence of a -define- command does not guarantee that all variables used had been given names.

To estimate the rate of usage of defined variables, we identified a set of 11 target variables and searched for references to them in each of the lessons (see Appendix II for

more details about the target set). The four possible outcomes were that: (a) each target variable used in the lesson had been defined, (b) no target variables used in the lesson had been defined, (c) some target variables used in the lesson had been defined and some had not, and (d) none of the 11 target variables had been used in the lesson. Table 3 reports the percentages of lessons at each site falling into categories (a), (b), (c), and (d).

The right-most column of Table 3 (% of lessons with no -define-s) is presented so that the reader can note the close congruence of those data with the information in the third column (% of lessons with no target variables defined). If one is interested only in which or how many lessons have no named variables, the data in the right-most column of Table 3 are far quicker to gather than those in the third column and hence provide essentially the same information more cheaply. All in all we have found the information in Table 3 to correlate well with more time-consuming personal observations. This approach seems adequate in most cases for making management decisions. For example, searching for -define- commands is a quick way for finding lessons which have not used named variables.

"Nameable" pieces of a lesson (in addition to variables) include -unit-s, -entry- points, and blocks. The names of these pieces may be chosen by the author, and his skill in assigning them may speed later identification and recall of their purpose. As part of this survey, we attempted to classify and categorize the patterns of authors' naming habits (e.g., numbers, abbreviations, acronyms), but failed to find any consistent or important patterns. Therefore, we have limited our discussion of on-line documentation to the topics of use of comments and use of named variables.

Table 3

Defined and Primitive Variables

Site	N[a]	% Lessons With All Target Variables Defined	% Lessons With No Target Variables Defined	% Lessons With Some Target Variables Defined	% Lessons With No Target Variables Observed	% Lessons With No -Define-s
CHA	37	0	86	14	0	86
ARI	20	95	5	0	0	10
MAX	7	100	0	0	0	0
ORL	16	25	31	38	6	26
ABE	22	0	55	45	0	45
SHP	66	27	5	44	24	38
NTC	15	0	100	0	0	93
N.I.	12	33	33	33	0	34

[a]N = total number of lessons examined at each site

Courseware Engineering

The construction features of a program which allow it to be efficient and effective in its medium constitute what we call courseware engineering for CBE. Efficiency is considered in terms of the extent to which the author's programming "style" interferes with his lesson's use by students or other authors. This concern can be thought of as a kind of "human engineering" applied to CBE courseware materials. Discussed in this section are several examples of poor courseware engineering habits which we observed and measured. These are: oversized lessons, poor implementation of drivers, and insufficient numbers of student restart points.

Lesson size. The amount of space that a lesson occupies is closely related to its usability. A large lesson, unless used by several students simultaneously, takes up more than its "fair share" of active computer memory (known on the PLATO system as Extended Core Storage or ECS). Because ECS is allocated among the various sites attached to the system, a single user of a large lesson at a given site may prevent other users at that site from accessing their desired lessons. Large lessons also are typically found to be more difficult to modify and troubleshoot. Therefore, the experienced, conscientious PLATO author generally tries to limit the size of his lessons to approximately 3000 computer words each and divides into two or more sections those lessons which are larger than about 5000 words.

Some lessons must, by their nature, fall outside the bounds of normal usage. However, among the large lessons produced by ARPA authors, only some of those developed at the Sheppard and San Diego sites (NTC and N.I.) were so monolithic that they could not be divided.

Lessons which are too small can also cause problems for the user. These hazards include a greater proportion of system

resources expended on "overhead" operations (e.g., condensing or disk accessing) and, if the same driver routine is used in each of several small lessons, a large amount of ECS may be taken up by duplicate copies of the driver's code. Some programs which present large amounts of text have utilized a small active driver routine designed to call up and display sections of text from a dataset file. Although such programs may be effective in reducing the amount of ECS in use at any one time, they may require more frequent disk accesses than can be easily supported by current system resources. In general, however, the problems associated with too-small lessons are not as troublesome as those which stem from excessive size, and no too-small lessons were observed in this study.

Table 4 portrays several perspectives on size of lessons at individual sites. A few extremely large programs can be tolerated at a site if their impact is diluted by the use of many other small lessons or if large lessons are used only during low usage periods. Therefore, to allow a few exceptional lessons to be "overlooked" in determining the range in lesson size, interquartile ranges were calculated. These are ranges for the "middle" group of lessons after the largest and smallest 25% have been discarded. All sizes are given in terms of the number of 60-bit computer "words" of ECS occupied by a lesson while in use.

Only a few sites reported having shortages of ECS because of their use of large lessons. Chanute AFB (Dallman, DeLeo, Main & Gillman, 1977) and the NPRDC sites, NTC and N.I., (Crawford, Hurlock, Padilla & Sassano, 1976) reported that they could not always use all their terminals because of ECS shortages during periods of peak computer usage (10 a.m. through 3 p.m. CST).

Table 4

Indicators of Size of TUTOR Programs Developed at ARPA Sites.

Site	N[a]	Mean	Median	S.D.	Full Range	Inter-quartile Range
CHA	37	4949	4979	2103	755-10375	3442-6140
ARI	20	3544	3990	1691	145-6240	2233-4774
MAX	7	3837	4056	1200	1688-5063	3562-4372
ORL	16	2492	2401	910	1246-4559	2013-2909
ABE	22	3821	3869	1177	1500-6714	3018-4260
SHP	66	5055	5117	1362	1500-7821	4061-6037
NTC	15	5412	5203	862	4171-8099	5091-5482
N.I.	12	3536	3691	2615	350-8235	1764-5009

[a] N = total number of lessons examined at each site

As seen in Table 4, the full or unattenuated range shows the CHA, NTC, and N.I. sites to be the ones having the largest single lessons. Since these also were the sites which reported the most difficulty with ECS shortages, it appears that the unattenuated range is an adequate measure for forecasting problems with long lessons (i.e., it probably is not necessary to compute interquartile ranges). Assuming that future sites would have similar distributions of lesson sizes and use patterns, it is apparent that site managers should

direct their attention toward trying to shorten the longest lessons. The data suggest, but do not prove, that dividing a few large lessons may significantly alleviate the difficulties.

Use of drivers. In order to make the most effective use of authors' time, various routines known as "drivers" have been developed (see Francis, 1976b, Appendix IV for an example). These routines are used to speed the construction of commonly used lesson segments such as tests, drills, data collection, etc. By using these pre-written routines, the author saves programming time when the lesson is created. By maintaining only a single copy of each routine, separated from the lessons employing it, the author also speeds revisions. For example, if the decision to "time" all students in drills is made after all the lessons are prepared, the authors who used a driver need make but one change--in the driver for the drills. Authors who have individually coded each drill or who have copied standard routines into each lesson must make the modifications in each individual lesson.

Two sites used drivers to great advantage. Several test drivers were developed by the MTC group and used by the Sheppard AFB staff. Authors at the San Diego NPRDC sites developed and successfully implemented drivers of their own.

The value of drivers was well-recognized among other ARPA PLATO sites, but their implementation was sometimes misunderstood, hence negating many potential benefits. For example, one site using MTC-supplied drivers copied the driver routines into EACH lesson, changing the names of units, -define-d variables, etc. Subsequently, when modifications were required, all lessons had to be individually revised with different variables having to be found and corrected in each lesson. If these routines had been used as originally intended (i.e.,

by maintaining only a single form of the code which could be accessed by other programs via a -use- command), it would only have been necessary to make changes in that original -use-d piece of coding.

Another site created drivers which employed very naive programming. A single multiple choice item, programmed using these drivers, consumed more than half of the student variables available to the programmer and several hundred words of ECS. Although the developer of these drivers hoped they would be used by a wide PLATO audience, it is not surprising that few authors made use of them.

A potential hazard of extensive use of driver routines is that they may be written in a too-general format. A developer may be tempted to add a variety of effort-saving features which may not be needed for all applications, hence increasing the overhead (in terms of ECS requirements) for those users who need only a subset of the driver's features. Thus, it is necessary to balance generality and potential for effort-savings of driver routines against the increased operating overhead they may require. Optimizing the design and application of drivers requires a degree of experience which few new authors will have.

Restart commands. -restart- commands are used to establish re-entry points so that a student can begin a new session at an appropriate place in his current lesson. Failure to insert these commands can cause problems for the student: if he leaves his lesson before completion or if a system interruption terminates his progress involuntarily, he will be forced to start again at the beginning of the lesson. This has been a source of difficulty and frustration for students in the past (see Klecka, 1977b).

The method we have used to measure the use of the -restart- feature involves counting the number of -restart- commands

in a lesson and dividing that total by the number of "blocks" of TUTOR source code in the lesson, a measure of the lesson's length. The resulting ratio is a more refined measure than the total -restart-s per lesson because it takes lesson size into account (i.e., it gives the number of -restart- commands per unit of lesson size).

Table 5 contains the raw counts of -restart- commands used at the various sites. Table 6 shows the number of -restart-s per block of source code.

Table 5
Number of -restart- Commands per Lesson

Site	N[a]	# Commands	% Lessons with Command
CHA	37	315	97%
ARI	20	21	60%
MAX	7	3	29%
ORL	17	43	88%
ABE	22	0	0%
SHP	66	205	29%
NTC	15	54	7%[b]
N.I.	12	1	8%[b]

[a] N = total number of lessons examined at each site

[b] Instead of using -restart- commands, these lessons relied on student variables to maintain the student's place in the lessons.

Table 6

Number of -restart- Commands per Block of Source Code

Site	N[a]	Mean	Median	S.D.	Full Range	Inter-quartile Range
CHA	37	0.49	0.4	0.38	0.00-1.60	0.21-0.64
ARI	20	0.09	0.06	0.12	0.00-0.44	0.00-0.10
MAX	7	0.04	0.00	0.08	0.00-0.20	0.00-0.00
ORL	16	0.24	0.29	0.18	0.00-0.50	0.08-0.36
ABE	22	0.00	0.00	0.00	0.00-0.00	0.00-0.00
SHP	66	0.18	0.00	0.35	0.00-1.80	0.00-0.24
NTC	15	0.30	0.00	1.16	0.00-4.50	0.00-0.00
N.I.	12	0.00	0.00	0.01	0.00-0.03	0.00-0.00

[a] N = total number of lessons examined at each site

Although feedback from some of the sites (as well as our general experience) indicates that -restart-s were used less frequently than might have been desired, we do not yet know what ratio of restarts per block should be taken as a minimum standard. Hopefully, this measurement technique can be applied to lessons for which independent measures of -restart- adequacy are available. If it is known, for example, that students rarely complain about having to repeat small

sections of material after unplanned interruptions in a given lesson, that lesson's -restart-s per block ratio may be a reasonable standard for other lessons in that subject area and for similar populations of users. However, since measures of storage space (e.g., blocks or units) would be strongly affected by authoring styles and demands of the material, a more appropriate measure might be -restart-s per display.

Measurement of Programming and Instructional Design Sophistication

An author's proficiency in effective and efficient use of the TUTOR language to program instructional lessons can be gauged by his use of certain commands (or language features such as "reserved words").

Reserved words. Reserved words (also known as system defined variables) are storage locations in which information relating to a student's progress through a lesson are automatically stored by the computer. An author can write programs in such a way that they utilize the information in the reserved words to individualize the instruction or collect student performance data.

A small group of the more than 100 reserved words in the TUTOR language (Avner, 1977) was selected as a target for computer-scanning of the ARPA lessons (see Appendix II for details). These reserved words fell into the following four categories of application:

1. area data, auto-collected by the PLATO IV system to give information on student performance without requiring the author to maintain a count of the number of questions answered correctly, the total number of questions, the total number of errors or requests for help, etc.

2. time data, to record the amount of time taken by a student to complete an instructional unit such as a drill, or the amount of time a student may choose to spend working practice problems.

3. feedback control information, to enable an author to prescribe feedback messages that take account of the number of times a student has attempted to answer that question.

4. answer judging information, to enable an author to prevent duplicate answers to multiple-answer questions, etc.

Use of reserved words varied widely in the programs examined in this study. Some lessons incorporated references to many reserved words, each several times; others made no references to the reserved words selected as targets for computer-scanning. The data for all elements of the target set were pooled and are reported in Table 7.

Table 7
Usage of Reserved Words

Site	N[a]	# Reserved Words	% Lessons with Reserved Words
CHA	37	53	49%
ARI	20	149	85%
MAX	7	69	100%
ORL	17	33	87%
ABE	22	16	73%
SHP	66	653	86%
NTC	18	18	11%
N.I.	12	19	58%

[a] N = total number of lessons examined at each site

Response handling and feedback. The use of certain commands and feedback after the judgment of student responses can indicate a degree of care in instructional design. The author of an instructional lesson may anticipate certain incorrect answers a student is likely to give. In such cases he or she might use a -wrong- command followed by a feedback message designed to correct the student's specific misunderstanding. A -no- command may be used to enable the author to provide general feedback to unanticipated incorrect responses (i.e., those which do not match a -wrong- command). An author who is less sophisticated in response handling might fail to include either -wrong- or -no- commands. In such cases the only corrective feedback the student receives is the "no" which is automatically provided by the system. Feedback delivered for specific responses could be of greater value in guiding the student toward the correct answer without blatantly giving it away (see Klecka, 1977a). In a similar vein, the use of multiple or sophisticated forms of the -answer- command enable an author to broaden the range of acceptable correct responses at relatively little cost. The provision of instructional "helps" via the -help- command may also serve to indicate care in instructional design.

Table 8 presents summary statistics for a variety of potentially-useful measures of the response handling and feedback characteristics of lessons produced at the ABE, SHP, and CHA sites. For purposes of interpretation it should be noted that the counts of -arrow- commands include any -arrow-s used for indicies and other applications which do not involve judgment of student responses. The counts shown for -wrong- commands also include instances of -wrongv-. Likewise, those given for -answer- commands include occurrences of -ansv- and --ansu- (see Appendix II for details).

Table 8

Measures of Response Handling, Feedback, and Remedial Assistance in SHP, ABE, and CHA[a] Lessons

Ratio	Site	Mean	Median	S.D.	Full Range	Interquartile Range
-wrong-/ -arrow-	SHP[b]	1.22	0.75	2.02	0.00 - 13.50	0.46 - 1.40
	ABE[c]	0.90	0.88	0.54	0.09 - 2.62	0.61 - 1.21
	CHA[d]	0.78	0.28	1.35	0.00 - 4.00	0.00 - 0.80
-no-/ -arrow-	SHP	0.76	0.76	0.60	0.00 - 3.62	0.50 - 0.94
	ABE	0.78	0.81	0.36	0.30 - 1.84	0.58 - 0.87
	CHA	0.36	0.39	0.26	0.06 - 0.71	0.10 - 0.46
(-wrong- + -no-)/ -arrow-	SHP	1.98	1.58	2.09	0.00 - 13.50	0.70 - 2.08
	ABE	1.70	1.62	0.76	0.40 - 4.13	1.36 - 2.06
	CHA	0.68	0.69	0.32	0.06 - 1.00	0.62 - 0.96
-answer-/ -arrow-	SHP	1.23	1.05	1.06	0.00 - 5.80	0.70 - 1.38
	ABE	0.74	0.81	0.32	0.07 - 1.13	0.67 - 1.00
	CHA	0.74	0.75	0.20	0.43 - 1.00	0.64 - 0.88
-help-/ -block-	SHP	0.10	0.03	0.18	0.00 - 0.87	0.00 - 0.13
	ABE	0.07	0.00	0.11	0.00 - 0.39	0.00 - 0.08
	CHA	0.04	0.00	0.07	0.00 - 0.19	0.00 - 0.00

[a] Data for the CHA lessons were collected while the lessons were in first draft form

[b] N=44 (i.e., 44 SHP lessons were examined)

[c] N=24

[d] N=8

Note. The various ratios used as measures of instructional design competence were computed within each lesson. The summary statistics reported describe the distribution of these ratios across the set of lessons at each site.

Of the variables measured (and reported in Table 8), there was remarkable similarity among the sites on -wrong-s/-arrow- and -help-s/block in spite of marked differences in the PLATO applications each employed. However, some statistically significant differences were found among the sites on the other measures. The ABE site's lessons had more -no-s/-arrow- (t_{30}=3.032, p=.005) than those from CHA. The CHA lessons also had significantly fewer (-wrong-+ -no-s)/-arrow- than ABE lessons (t_{30}=3.657, p=.005) or SHP lessons (t_{50}=3.883, p=.0003). Finally, the SHP lessons had a significantly higher -answer-/-arrow- ratio than either the ABE lessons (t_{66}=2.838, p=.0066) or the CHA lessons (t_{50}=2.804, p=.0076). (Probability values for the t-statistics resulting from the last three comparisons were estimated by Welch's correction since groups had unequal variances).

Because the student populations and instructional strategies employed at each site differed widely, it would be inappropriate to assume that these data alone indicate superiority of one set of lessons over another. For example, it is possible that the context in which the CHA lessons were used made it unnecessary to provide for a broad range of acceptable correct responses or diverse anticipated incorrect responses. It does appear, however, that measures amenable to computer-scanning techniques are sensitive to differences among sets of lessons and that they are potentially useful to managers of CBE projects.

DISCUSSION AND CONCLUSIONS

Specific Findings

Documentation. The lessons in the target group of ARPA-sponsored TUTOR programs were not thoroughly documented. This conclusion follows both from measurements we have taken (see Tables 2 and 3) and subjective judgment based on our experience as "caretakers" of lessons following project completion.

One reason for the meager levels of documentation is that most of the ARPA PLATO authors had no previous programming experience or training. The importance of documentation is stressed in most formal computer science training programs, and these attitudes are reinforced in work environments. Lacking prior experience, the ARPA authors had not previously "internalized" the value of well-documented programming. Furthermore, many ARPA projects were considered to be short-lived "experimental" efforts whose programs would not require maintenance after the project had ended (hence lowering the perceived importance of documentation). Even for those authors who did develop the habit of inserting comments in their more recent lessons, the burden of going back to add comments to earlier work may have been felt as being too great to undertake. In general, there were few environmental rewards for including ample documentation and few punishments for not.

Courseware engineering. The issue of lesson size as it relates to the periodic shortages in ECS experienced during the five year history of the ARPA/PLATO projects has an interesting parallel to the energy crisis in the U. S. During periods when ECS is plentiful in relation to demand, there are few "punishments" for writing excessively large lessons. Problems do occur, however, when demand increases to exceed a site's ECS allocation. Unfortunately, the obvious remedy of reprogramming lessons to reduce their size is often viewed

as too "painful" or expensive. Hence some people argue for conservation of resources (i.e., keeping lesson size small) while others advocate expanding resources (buying or building more memory space).

There is no doubt that lazy programming habits wasted ECS, and that some ARPA authors were adamant about not dividing oversize lessons. There is also no doubt that there are some legitimate needs for lessons as large as or larger than the current limits. What is most surprising to us is the apparent ineffectiveness of incentives for encouraging good programming habits. For example, a PLATO site with more than 12 terminals typically "owns" and manages its own ECS pool (size based on number of terminals). By limiting (or reducing) the size of its lessons, a site gains additional flexibility and the ability to use all of its terminals simultaneously for a greater variety of activities. By failing to set or enforce lesson size limits, the site is forced to let some of its terminals remain idle or to implement rigid schedules of use. In our experience, this penalty had a very small effect for motivating staff to divide large lessons; they preferred to let some terminals go unused.

The problems with "driver" programs noted earlier seem largely to be the result of mistaken impressions regarding what drivers do, how they operate, and which resources are scarce vs. which are plentiful. A description in "aids" or a chapter in an off-line manual could indicate the characteristics of a well-constructed driver and thereby attempt to eliminate the inefficient practices identified in this study."

The data presented regarding the use of -restart-s in instructional lessons (see Table 6) lead to the conclusion that more consistent usage would have been desirable. One possible explanation for this is that -restart-s are often left out until after students begin to use the lessons.

Authors (since they are less affected by unplanned interruptions or the necessity to attend other classes, etc.) may not notice the absence of -restart-s until actual students begin to complain. Even student tryouts of new materials may not alert the author to cases of insufficient numbers of -restart-s, because such trials are usually completed in one sitting. Some authors may not have added -restart-s even after lessons were put into use, either because of lack of understanding of their importance or a belief that adding them was a difficult task. We believe that these notions can be dispelled through proper training.

Reserved words, response handling, and feedback. Commands and language features discussed earlier under these headings can be economically described as "interaction commands". The potential for extracting useful data via computer-scanning seems especially great for commands in this category. Because of the large numbers of interaction commands present in most lessons, their ratios tend to be stable from one lesson to another (see Table 8). This stability suggests that it may be possible to describe minimal standards for frequency of interaction commands which can serve as guides for new authors or as measures of how well a given lesson is adapted to an interactive medium such as CBE.

Although this investigation measured ratios of commands present in lessons' TUTOR coding (indicative of potential interactions and feedback), we were not able to acquire sufficiently well-matched data to determine ratios of those commands encountered by students during study of the lessons (i.e., actual interactions and feedback). Such "execution time" data could validate analyses of coding used in lessons and aid their interpretation. Techniques for assessing the meaning and usefulness of student interaction data are more advanced at this time than those described here for program

command analysis (see Francis & Weaver, 1977), but the potential of the latter should be further explored and exploited.

Comments on Management Uses of the Computer Searching Technique

This report demonstrates that several categories of data useful in operating a PLATO CBE site can be gathered efficiently by means of various features of the PLATO system itself (particularly its string-searching capability). Levels of documentation and -restart- use can be monitored easily with these techniques, and the potential for study of interaction command use has been noted. This report also suggests ways for minimizing the time and expense for data collection. For example, it appears that monitoring the size of only the largest lessons may be sufficient for avoiding problems of long lessons and that merely checking for the presence of -define- commands may be an adequate measure of use of named variables.

Although most of these forms of data can be easily collected by clerks having a minimal amount of training, we feel that project managers may prefer to make these checks themselves on an informal but regular basis. There is a need for personal experience before many of these procedures are internalized and managers are in the best position to make decisions about how to use the data derived from these techniques. We recognize (and have experienced) the difficulty of convincing authors of the need for anything that causes even slightly more work in the short term when payoffs are greatest in the long term (for an example, see the discussion regarding documentation). We believe, however, that skillful use of computer-scanning techniques can give managers an empirical basis for establishing guidelines and assessing authors' adherence to them.

REFERENCES

Avner, E. PLATO user's memo: Summary of TUTOR commands and system variables (6th edition). Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, September 1977.

Crawford, A. M., Hurlock, R. E., Padilla, R., and Sassano, A. Low cost part-task training using interactive computer graphics for simulation of operational equipment. San Diego: Navy Personnel Research and Development Center, 1976.

Dallman, B. E., DeLeo, P. J., Main, P. S., & Gillman, D. C. Evaluation of PLATO IV in vehicle maintenance training. (AFHRL -TR- 77-59) Lowry Air Force Base, Colorado 80230: Air Force Human Resources Laboratory, Technical Training Division, 1977.

Francis, L. PLATO IV terminal peripheral devices. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1976a.

Francis, L. The TUTOR training course: Lessons learned. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1976b.

Francis, L. & Weaver, T. Analysis of student interaction data in computer-based education. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1977.

Klecka, J. A. An overview of chanute lessons. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1977a.

Klecka, J. A. Three aspects of PLATO use at Chanute AFB: CBE production techniques, Computer-aided management, and Formative development of CBE lessons. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1977b.

Lyman, E. R. PLATO highlights. Urbana, Ill.: University of Illinois, Computer-based Education Laboratory, 1977.

Sherwood, B. The TUTOR language. (#76360692) Minneapolis: Control Data Education Company, 1977.

Wood, N. The PLATO system. Urbana, Ill.: University of Illinois, Computer-based Education Research Laboratory, 1975.

APPENDIX I

Sites Selected for this Study

School of Applied Aerospace Sciences, Chanute Air Force Base, Illinois (CHA)

U.S. Army Ordnance Center and School, Aberdeen Proving Ground, Maryland (ABE)

School of Health Care Sciences, Sheppard Air Force Base, Texas (SHP)

Navy Personnel Research and Development Center, San Diego, California

1. NTC - Navy Training Center (San Diego)
2. N.I. - Navy Training Center (North Island)

Naval Training Equipment Center, Orlando, Florida (ORL)

Air University, Maxwell Air Force Base, Alabama (MAX)

Army Research Institute, Washington, D.C. (ARI)

Of the above sites, Aberdeen, Orlando, and Maxwell were no longer subscribing to PLATO service at the time of the preparation of this report.

APPENDIX II

Character Strings used as Targets for Computer-Scanning

Object	Character Strings
Defines [a]	define, n2, v2, nc2, vc2
Comments	$, c , *
Restarts/ Lesson	restart
Reserved words [b]	judged, clock, aokist, ntries, jcount, anscnt
Anticipated/ unanticipated responses	wrong, no, arrow, answer, help, wrongv, ansv, ansu

[a] note that searches for the string "n2" will also result in matches for the n20-n29 variables. The rationale for this choice of target variables is as follows: n1 would have located too many variables for the purpose of this study (n1, n10-n19, plus n100-n150). Also, the number of authors using variables n100-n150 would be too small and the TUTOR training given the ARPA authors suggested that the variables n1-n9 be used as "scratch" variables which would not need to be -define-d. (Since they require only two keystrokes, that set is preferred.) The last consideration was that we needed a set of variables that most people would be likely to have used (hence n80-n89, etc. were not selected on the assumption that lower-numbered

variables would have been used first). Considering these criteria, the best choice was n2 plus n20-n29. For completeness, the "v", "vc", and "nc" forms were also targets for the search.

[b] information contained in the selected reserved words:

judged	number of times student responses to questions have been judged
clock	amount of time spent in lesson or section of lesson
aokist	number of questions answered correctly on first try
ntries	number of attempts at current question
jcount	number of internal 6-bit character codes in student response
anscnt	number of answer-judging commands encountered before matching the student's response